**ABSTRACT**

Text summarization is an automated process that aims to generate a concise summary comprising important sentences, encompassing all relevant information from the original document. Two primary approaches in summarization, namely Extractive and Abstractive, have been observed based on the summary outcomes. While Extractive summarization has reached a certain level of maturity, current research endeavors have shifted focus towards Abstractive summarization and real-time summarization.

Over the past decade, there have been significant advancements in the acquisition of datasets, methods, and techniques in the field. The exponential growth of data on the Internet necessitates a solution that can effectively transform vast amounts of raw information into meaningful knowledge that can be comprehended by the human brain. Text summarization has emerged as a common technique in research to address the challenges posed by the enormous volume of data.

The analysis results provide an in-depth understanding of the ongoing research trends in text summarization. They offer references to publicly available datasets, discuss preprocessing techniques and employed features, and describe the commonly used techniques and methods employed by researchers for comparison and method development purposes. Moreover, this research paper concludes by highlighting several recommendations concerning the opportunities and challenges associated with text summarization research.

To further enhance the field of text summarization, it is crucial to explore approaches that can handle multi-document summarization, tackle bias and subjectivity, and preserve the original intent of the source text. Future research should also focus on refining existing methods, exploring innovative techniques, and leveraging advancements in natural language processing and machine learning algorithms. By addressing these challenges and embracing opportunities, the field of text summarization can continue to evolve and provide valuable solutions for information extraction and understanding.

**CHAPTER 1**

**INTRODUCTION**

With the rapid increase in web-based information available in various formats like text, video, and images, individuals often struggle to find relevant information of interest. When users query for information online, they are bombarded with numerous search results that may not be directly related to their specific needs. This information overload creates a time-consuming and effort-intensive task of sifting through documents to find the desired content. Automatic text summarization emerges as a valuable solution to address this problem.

Automatic summarization plays a crucial role in condensing source documents into concise and meaningful content that captures the main ideas without altering the information. By providing an effective summary, users can quickly grasp the key points without having to go through the entire document, ultimately saving time and effort. The text summarization process involves three steps: analysis, transformation, and synthesis. Analysis involves examining the source text and selecting relevant attributes. The transformation step processes the analyzed information, and finally, the synthesis step generates a summary representation.

Text summarization approaches are broadly categorized into extractive and abstractive summarization. Extractive summarization involves extracting important sentences or phrases from the source documents and grouping them together to create a summary while preserving the original text. On the other hand, abstractive summarization focuses on understanding the source text using linguistic methods to interpret and analyze its content. The goal of abstractive summarization is to generate a concise summary that conveys the information in a more generalized manner.

**1.1 PROBLEM DESCRIPTION AND OVERVIEW**

The internet offers a vast amount of information, but it has created a strange situation. People have access to so much data, yet they still crave wisdom. Keeping up with the billions of articles produced daily is challenging. To address this issue, we propose developing a Text Summarizer using specific methods.

**1.2 OBJECTIVE**

The aim of text summarization is to condense a document to its key points. This can be done by humans or using algorithms. The goal is to create a brief summary that captures the main ideas of the original text. Text summarization saves time when reading long papers, such as research articles, by providing a concise summary without leaving out important information.

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 AUTOMATIC TEXT SUMMARIZATION**

Authors: Ujjwal Rani, Karambir Bidhan

Content summarization involves condensing a source document into a concise form that captures the main ideas. There are two approaches to text summarization: abstractive and extractive. Abstractive summarization uses natural language processing techniques to generate summaries, while extractive summarization relies on statistical, linguistic, and heuristic methods to select and rank sentences. Various methods have been developed for text summarization in different languages. This paper focuses on exploring abstractive summarization techniques.

**Abstractive Summarization**

 The goal is to create a brief summary that captures the main ideas of the source text. In abstractive summarization, new phrases and sentences can be included in the summary that are not present in the original text. Abstractive summarization approaches are rooted in a structure-based approach.

* **Structure Based Approach**

Structure-based methods in summarization involve utilizing prior knowledge and mental constructs such as templates, extraction rules, and alternative structures like trees, headings, and graphs to capture the most important information.

* **Tree based**

The tree-based approach in summarization involves using a dependency tree to analyze the content of a document. This method relies on language generators to generate accurate summaries. In the first step, the approach creates a dependency tree by dividing sentences into chunks. Then, it determines the importance of each part of the tree. Subtrees from different sections are identified and added to the dependency tree to enhance its structure. Finally, predefined elements are removed from the tree to refine the summary.

* **Rule based**

Information extraction rules produce output, and the content selection module selects the best option based on categorization to provide a suitable response. The primary objective of this method is to generate a summary that is more informative than the existing one.

* **Lead and body phrase method**

The lead and body phrase method focuses on identifying common phrases in the lead and body sections of the text. It then generates a summary by revising sentences through the insertion and substitution of these phrases. This approach is inspired by systems that combine sentences. However, a limitation of this method is that parsing errors can negatively impact the structure of the summary sentences, resulting in grammatical errors and redundancy.

* **Ontology based**

Ontology-based summarization is a new area within information extraction. It is inspired by the concept of text summarization in natural language processing. In ontology summarization, the goal is to extract relevant knowledge from an ontology and create a condensed version tailored to a specific user or task.

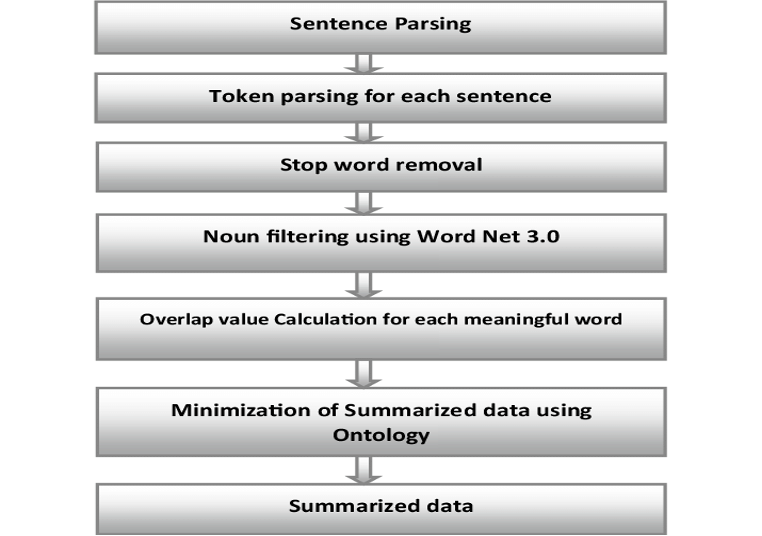


Fig 2.1 Ontology Based Summarization

* **Graph based**

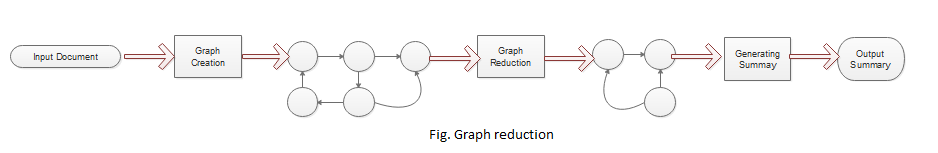
 Text summarization can be achieved using a graph-based approach, specifically through an unsupervised technique. In this technique, sentences or words are ranked based on their position within a graph. The primary objective of the graph-based method is to identify the most important sentences within a document. The graph represents the document or sentences as nodes, while edges connect nodes that share common information. The weightage or importance of each sentence is determined by assigning initial weights to the nodes of the graph.

Fig 2.2 Graph Reduction

* **Information item (INIT) based**

This technique utilizes abstract representation to generate a precise summary of the document, rather than directly using sentences from the source document. The fundamental unit of abstract representation is the information item (INIT), which captures relevant information. The resulting summary is concise, intelligent, informative, and avoids excessive repetition. At the initial stage, a parser assists in the syntactical analysis of the document by retrieving INIT and forming subject-verb-object (SVO) structures. A language generator is employed to produce sentences. The sentences are then ranked based on their frequency scores, and only the high-ranking sentences are included in the summary. However, this approach is prone to grammatical errors and may struggle with generating meaningful sentences. Furthermore, the linguistic quality of the summary produced is often low due to incorrect parses.

**2.2 NATURAL LANGUAGE PROCESSING (NLP) BASED TEXT SUMMARIZATION**

Author: Ishitva Awasthi, Kuntal Gupta, Prabjot Singh Bhogal

In practical terms, Natural Language Processing (NLP) involves the use of computers to perform various tasks, including text summarization, parsing, entity recognition, and speech recognition. Text summarization in NLP is commonly achieved by extracting the main paragraphs or key information from a given text.

Text summarization is a process within NLP where artificial intelligence or machine-learning algorithms analyze text to infer its main topic or subject. Unlike deep learning, text summarization is focused on machine-based approaches, which are well-suited for generating quick outputs when dealing with short-text processing requirements.

Text summarization using NLP is a fundamental process for condensing text content by highlighting key features and presenting them in a concise manner. NLP, in general, refers to the use of computer systems to enhance the interaction between machines and humans through natural language understanding and processing.

* **Supervised Text Summarization:**

Supervised text summarization relies on having labeled training data, where each document is accompanied by its corresponding summary. This method involves training a machine learning model, such as a sequence-to-sequence model, using this labeled data. The model is trained to understand the relationship between the input document and its corresponding summary by learning from the provided examples. During training, the model is exposed to both the original document and the desired summary, allowing it to learn how to generate concise summaries by predicting the most suitable sequence of words. However, this approach necessitates a significant amount of annotated data, and the quality of the generated summaries depends on the quality and representativeness of the training data used.

* **Unsupervised Text Summarization:**

Unsupervised text summarization, in contrast to supervised methods, does not depend on labeled data. Its objective is to automatically extract the most essential information from the source document without predefined summaries. There are various approaches used in unsupervised summarization:

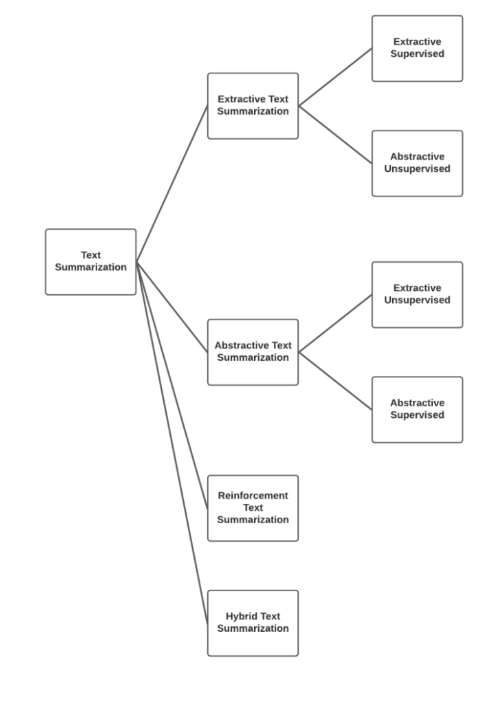


Fig 2.3. Supervised & unsupervised summarization

**2.3 EXTRACTIVE TEXT SUMMARIZATION**

AUTHORS: Arpita Sahoo, Dr.Ajit Kumar Nayak

* **Summarization attributes of extractive**

In extractive text summarization, the focus is on identifying relevant sentences from the original document and including them in the summary. This approach involves analyzing the document to determine the importance of each sentence based on certain features or criteria. The sentences that meet the criteria are then selected and combined to form the summary. This method aims to extract and present the most crucial information from the source document without modifying or rephrasing the sentences.

* **Cue words features:**

Cue words or clauses are groups of words that appear around key words such as "summary," "reflects," "concludes," "purpose," "because," and so on. These cue words indicate the overall content of the document and can serve as indicators for determining which sentences should be included in the summary. By identifying these cue words or clauses, one can gain insights into the main ideas and important information contained within the document, allowing for effective selection of sentences for the summary.

* **Keyword features:**

A crucial feature of extractive summarization is the keyword feature. In this approach, sentences that contain a significant number of keywords are considered for inclusion in the summary. Keywords are specific words or phrases that carry substantial meaning and represent important concepts or ideas within the document. By identifying and prioritizing sentences that contain a high concentration of keywords, extractive summarization aims to capture the essential information and convey it in a concise summary.

* **Title word feature:**

In extractive summarization, the sentences that include the title or title-related terms are regarded as important and necessary sentences. These sentences are considered significant because they directly relate to the main subject or topic of the document. Including these title-related sentences in the final summary helps ensure that the summary captures the core essence of the document and provides a clear representation of its main focus. By incorporating these important sentences, the summary can effectively convey the key information to the readers.

* **Topic Modelling:**

Topic modeling techniques, like Latent Dirichlet Allocation (LDA) or Non-Negative Matrix Factorization (NMF), can be employed to uncover the latent topics or themes present in a document. These techniques help in identifying the underlying patterns and topics by analyzing the distribution of words and their relationships within the text. By applying topic modeling, the summarization process can be guided by identifying the most relevant and representative sentences for each topic. This approach enables the extraction of key information from different aspects of the document, resulting in a more comprehensive and informative summary.

* **Sentence Similarity**:

Assessing the similarity between sentences plays a crucial role in extractive summarization. It helps in determining the redundancy or similarity of sentences within a document, enabling the identification of repetitive information. By measuring the similarity between sentences, redundant or similar sentences can be identified and potentially excluded from the summary. This process helps in improving the overall quality of the summary by reducing redundancy and ensuring that each included sentence provides unique and valuable information.

* **Sentence length feature**:

In text summarization, the length of sentences is an important consideration. The goal is to reduce the length of sentences while preserving their content and meaning. This is done to create concise summaries that capture the essential information from the original document. By condensing the sentences without altering their intended message, the summarization method aims to provide a shorter version of the text that is easier to comprehend and digest.

* **Upper case word feature**

In the process of text summarization, it can be beneficial to include words that are written in all uppercase or are considered abbreviations. These words often carry specific significance or represent key concepts in the document. By including them in the summary, the summarization method ensures that important information and specific terminology are retained, allowing the summary to accurately convey the main ideas of the original document.

* **Term frequency:**

In text summarization, the TF-IDF (Term Frequency-Inverse Document Frequency) method can be employed to calculate the frequency of each word within sentences. This technique assigns a higher weight to words that have a higher frequency within a particular sentence and are relatively rare across the entire document collection. By considering the TF-IDF scores, the summarization method can prioritize words that occur frequently within sentences and are potentially more significant in conveying the overall meaning of the document. Including such high-frequency words in the summary can help ensure that the most important and representative information is captured.

* **Limitation of extractive method**
* **Redundancy**

Extractive summarization methods indeed focus on selecting important and relevant sentences, but redundancy can be a challenge. It is possible that similar information is repeated across multiple selected sentences, which can hinder the overall conciseness of the summary and may not effectively capture the most essential information.

To address this issue, various techniques can be employed. One approach is to assess the similarity between selected sentences and identify redundant information. By measuring the similarity, redundant sentences can be identified and potentially removed to enhance the overall coherence and conciseness of the summary. Additionally, incorporating sentence compression or fusion techniques can help condense redundant information while preserving the key ideas.

Overall, minimizing redundancy in extractive summarization is crucial to create concise and informative summaries that efficiently capture the main points of the source document.

* **Lack of Coherence**

One of the challenges in extractive summarization is maintaining the coherence and flow of the summary. Since extractive methods rely on selecting and stitching together existing sentences, there can be a lack of smooth transitions between the sentences, resulting in a summary that may not read as a cohesive text.

To address this issue, additional post-processing steps can be applied to improve the coherence of the summary. Techniques such as sentence reordering, rephrasing, and using connecting phrases or discourse markers can help improve the overall flow and readability of the summary. Additionally, incorporating language generation models or abstractive methods can be explored to generate more coherent and cohesive summaries by paraphrasing and rephrasing the extracted sentences.

It is worth noting that achieving perfect coherence in extractive summarization can be challenging, as the method heavily relies on the available sentences. However, by employing various post-processing techniques and considering alternative summarization approaches, the coherence of the summary can be improved to provide a more cohesive reading experience.

* **Information Loss**:

When using extractive summarization, important details can be missed if they are spread across multiple sentences or not included in the selected sentences. This means that the summary may not fully capture the complete context or subtle details present in the original text.

* **Sensitivity to Sentence Order**

In extractive summarization, sentences are often chosen individually without considering their overall order in the summary.

* **Limited Adaptability to Diverse Texts**:

Extractive methods may face difficulties when dealing with complex texts like scientific papers, legal documents, or highly technical writings. The focus on analyzing and ranking sentences at the individual level may not accurately capture the main ideas or concepts present in such texts..

**2.4 TEXT SUMMARIZATION: AN OVERVIEW**

Author: Mr.S.A.Babar

* **Main steps for text summarization**:

The process of summarizing documents typically involves three main steps: identifying the topic, interpreting the content, and generating the summary.

* **Topic Identification:**

To identify the most important information in the text, various techniques are employed, such as analyzing the position of phrases, identifying cue phrases, and examining word frequency. Among these techniques, those based on the position of phrases tend to be the most effective for topic identification.

* **Interpretation:**

In the interpretation step of abstract summaries, various subjects or topics are combined to create a cohesive and overarching content. This process involves integrating different aspects or themes from the original text to form a comprehensive and generalized summary.

* **Summary Generation:**

In the interpretation step, the system employs a text generation method. This technique involves generating new text based on the combined subjects and themes identified in the previous step. The system uses algorithms or models to generate coherent and meaningful sentences that capture the essence of the original text in a concise manner.

* **Text summarization history:**

Extractive summarization methods primarily rely on scoring sentences within the source document. During the 1970s, researchers introduced frequency-based methods as a statistical approach to text summarization. These methods involved analyzing the frequency of important terms within sentences to determine their importance. Early systems such as AutoSummarize and Coh-Metrix were developed during this time, utilizing term frequency analysis to identify significant sentences for summarization.

* **Methods of extractive summerization**
* **TF-IDF**

TF-IDF, which stands for Term Frequency-Inverse Document Frequency, is a technique used to measure the relevance of a word in a text document. It calculates a numerical value that increases as the number of times a word appears in the text increases. TF-IDF takes into account both the frequency of a word within a specific document (term frequency) and its importance in the overall corpus of documents (inverse document frequency). This approach helps identify words that are more significant and informative within a particular document compared to others.

**Terminologies**:

* **Term Frequency:**

In a document (d), the frequency (tf) of a word (t) represents the number of times that word appears in the document. The formula for calculating the term frequency is the count of the word (t) in the document (d) divided by the total number of words in the document (d). The term frequency helps indicate the significance or relevance of a word within a specific document. It does not consider the order of the words in the document; it focuses solely on the occurrence and frequency of the word.

tf(t,d) = count of t in d / number of words in d

* **Document Frequency**

This test is very similar to TF (Term Frequency) but with one difference. In the entire collection of documents (corpus), DF (Document Frequency) represents the number of occurrences of a term (t) in the document set (N), whereas TF is the frequency count of the term (t) in a specific document (d). DF provides information about how commonly a term appears across the entire document set, whereas TF focuses on the frequency of a term within a single document.

df(t) = occurrence of t in documents

**2.5 TEXT SUMMARIZATION TECHNIQUES**

Author: Mehdi Allahyari, Seyedamin Pouriyeh, Mehdi Assefi

* **Latent Semantic Analysis (LSA)**

Latent Semantic Analysis (LSA) is a statistical technique used in text analysis and natural language processing. It can also be applied to tasks involving text summarization. LSA works by analyzing the patterns of word co-occurrence in order to uncover underlying semantic connections between words and documents. By doing so, it aims to capture the latent or hidden meaning present in the text data.

**Benefits of LSA**

LSA offers several benefits and applications in text analysis:

1. Dimensionality Reduction: LSA helps in reducing the dimensionality of large text-based datasets. By representing documents and words in a lower-dimensional space, it simplifies the analysis and makes it more computationally efficient.
2. Topic Understanding: LSA provides insights into the latent topics encoded within a text corpus. By identifying patterns and associations between words and documents, it helps in understanding the underlying themes or topics present in the data.

3. Word Association Analysis: LSA allows for the examination of word associations within a text corpus. It can uncover relationships and connections between words, providing valuable information about how words are semantically related.

4. Term Relations: LSA can identify relations between terms by analyzing their co-occurrence patterns. It helps in understanding the semantic relationships and connections between terms, enabling better information retrieval and retrieval of related terms.

5. Prior Art Searches: LSA has been utilized in assisting prior art searches for patents. By analyzing the relationships between terms and documents, it helps in identifying relevant prior art and determining the novelty of inventions.

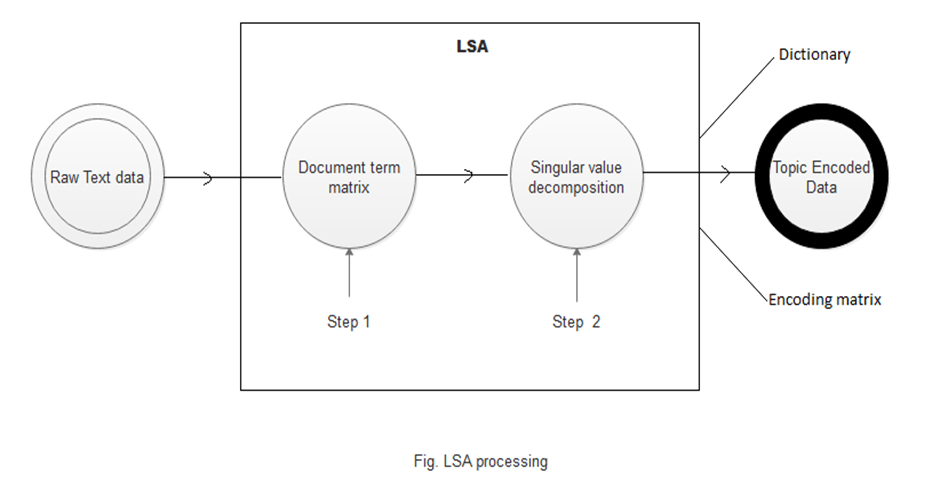
Overall, LSA is a versatile technique that aids in dimensionality reduction, topic understanding, word association analysis, term relations, and assisting in prior art searches for patents.

Fig 2.4 LSA Processing

* **Limitations**

1. Lack of Word Order and Syntactic Information: LSA overlooks the information related to word order, syntactic relationships, and morphologies present in the text. This can be a limitation because these aspects play a crucial role in understanding the meaning of words and texts. Without considering these elements, LSA may not capture the full linguistic nuances and context.

2. Limited Use of World Knowledge: LSA solely relies on the information available within the input documents and does not incorporate external world knowledge. This can be a drawback as incorporating external knowledge, such as background information or domain-specific knowledge, can enhance the accuracy and contextual understanding of the analysis.

3. Performance Challenges with Large and Inhomogeneous Data: LSA's performance tends to decline when dealing with larger and more diverse datasets. This decline in performance is mainly attributed to the Singular Value Decomposition (SVD) algorithm, which is computationally intensive and can struggle with complex and heterogeneous data. As a result, the accuracy and efficiency of LSA may decrease as the data size and heterogeneity increase.

In summary, the limitations of LSA include the neglect of word order and syntactic information, the limited utilization of external world knowledge, and challenges in handling larger and more diverse datasets that can impact the algorithm's performance.

**5.6 AUTOMATIC MULTIPLE DOCUMENTS TEXT SUMMARIZATION**

Author: Md. Majharul Haque1, Suraiya Pervin1, and Zerina Begum2

* **Time based method**

A time-based approach to text summarization focuses on understanding the temporal aspects of information in a document and creating a summary based on it. The goal is to either gather the most important and relevant information from different time periods or generate a summary that emphasizes current or up-to-date information. Here is a simplified explanation of how a time-based text summarization technique could work:

1. Preparing the Document: The document is processed to remove unnecessary elements like stop words or punctuation. It is also divided into sentences or smaller parts for analysis.

2. Extracting Timestamps: If the document contains timestamps or date information, they are extracted using techniques like regular expressions or natural language processing.

3. Selecting a Time Window: A specific time window is chosen to define the period of interest for the summary. This window can be fixed, such as the last month or year, or dynamically determined based on the timestamps in the document.

4. Scoring Relevance: Each sentence or unit of text is assigned a score based on its relevance to the selected time window. Sentences that fall within the time window or have timestamps close to it are considered more relevant. Different algorithms, like keyword matching or semantic similarity, can be used for this scoring.

5. Ranking Importance: In addition to relevance, sentences can be ranked based on their importance or significance within the time window. Factors like the number of occurrences, popularity, or source credibility may determine the importance ranking.

6. Generating the Summary: The top-ranked and most relevant sentences from the time window are chosen to create the summary. The length of the summary can be predetermined or adjusted based on the available content and desired level of detail.

7. Post-processing: The generated summary may undergo further refinement to improve coherence and readability. Techniques like sentence compression, removing redundant information, or applying linguistic rules can be used for post-processing.

A time-based text summarization approach involves analyzing the temporal aspects of information, selecting relevant sentences from a specific time window, and generating a concise summary that captures important details.

**5.7 IMPROVING PERFORMANCE OF TEXT SUMMARIZATION**

Author: S.A.Babara,Pallavi D.Patilb

* **Fuzzy logic scoring**

In text summarization, fuzzy logic scoring is a method that applies principles and techniques of fuzzy logic to assign scores or weights to sentences for determining their importance or relevance in the summary. Fuzzy logic allows for the consideration of different levels of truth and membership in linguistic variables, which helps in dealing with the inherent ambiguity and imprecision present in natural language. The process of using fuzzy logic scoring in text summarization involves the following steps:

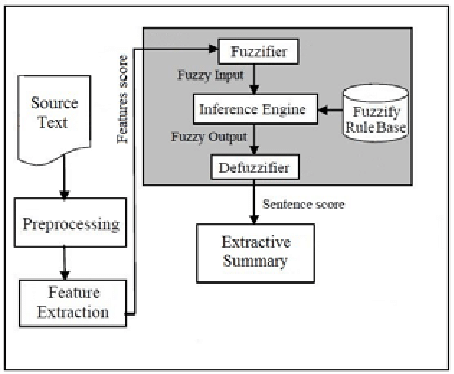
****

Fig 2.5 Fuzzy Logic Diagram

**5.8 AUTOMATED TEXT SUMMARIZATION**

Author: Eduard Hovy and ChinYew Lin

* **Cluster based method**

In this approach, the content of a document is analyzed by extracting and representing its semantic structure using triplets consisting of subjects, verbs, and objects associated with each sentence. These triplets are then clustered based on their similarity, grouping together those with similar information. The triplets statements serve as the fundamental units in the summarization process. By identifying and clustering similar triplets, redundant information can be minimized. This allows for the construction of a summary using a sequence of sentences that are related to the computed clusters.

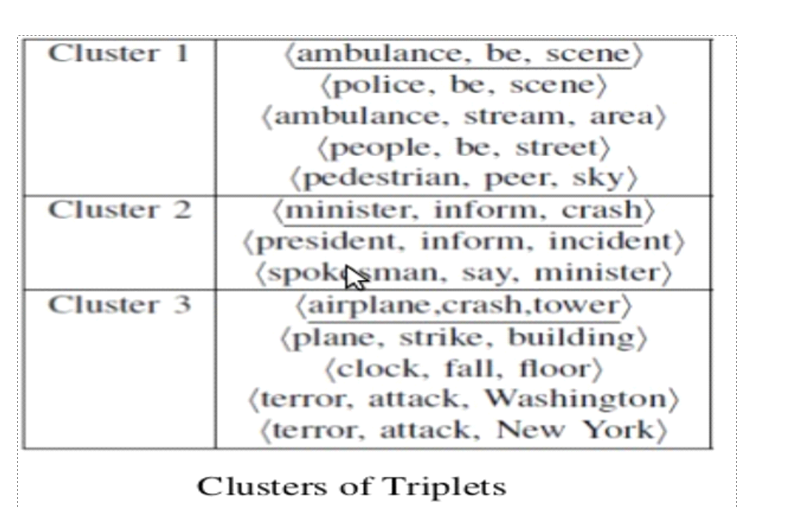


Fig 2.6 Clusters of Triplets

* **MACHINE LEARNING**
* A training documents set along with its summaries in extractive form are given as input to the training stage.
* Machine learning approach classified as supervised, unsupervised or semi-supervised.
* Bayesian rule is used to statistically determine the classification probabilities

P (s∈&lt;S | F1, F2, ..., FN)= P (F1, F2, ..., FN | s∈S) \*P (s∈S) / P (F1, F2,...,FN)

In this framework, each sentence S from the document collection is considered, and a set of features F1, F2,...,FN is used for classification. The goal is to determine whether each sentence should be included in the summary or not. The summary S to be produced is a collection of sentences that are deemed to be summary sentences based on the given features. The probability P(s ∈ <S | F1, F2,...,FN) represents the likelihood that a sentence s belongs to the summary based on the observed features. By calculating these probabilities, sentences can be classified as either summary sentences or non-summary sentences, allowing for the construction of an effective summary.

**5.9 AUTOMATIC TEXT SUMMARIZATION: A COMPREHENSIVE SURVEY**

Author: Wafaa S. El-Kassas

* **Query based method**

The query-based text summarizer utilizes a graph structure to analyze the relationships between sentences and words. It leverages various methods and algorithms that incorporate statistical and linguistic techniques. In the past, different approaches have been used individually, but to enhance the effectiveness of the summarizer, a combination of these techniques is necessary. By integrating multiple approaches, the summarizer can achieve improved efficiency and generate more accurate and informative summaries.

**Algorithm:**

1. Arrange sentences based on their score.

2. Include sentences from the title or topic of the document.

3. Include the first-level heading as part of the summary.

4. Check if the summary size limit is not exceeded.

5. Add the highest scored statement to the summary.

6. Include the structural context of the statement if it has not already been added.

7. Add the highest-level heading above the extracted text (let's refer to this heading as 'h').

8. Add the heading preceding 'h' at the same level.

9. Add the heading following 'h' at the same level.

10. Repeat steps 7, 8, and 9 for the next highest-level headings.

11. Continue these steps until the while loop is terminated.

This algorithm outlines the process of generating a summary by arranging sentences based on their scores and incorporating relevant headings and structural context. The loop continues until the summary size limit is reached or no further headings are available.

**2.10 A SURVEY OF TEXT SUMMARIZATION EXTRACTIVE TECHNIQUES**

Author: Vishal Gupta

* **Neural Network**
* A neural network is a network of interconnected artificial neurons that utilize a numerical model for processing data.
* In the context of text summarization, the approach involves training the neural network to recognize the type of sentences that should be included in the summary.
* The neural network is trained using sentences from a test paragraph, where each sentence is evaluated to determine if it should be included in the summary or not.
* Once the neural network is trained, small weights are pruned to remove less important features.
* The summary is then generated by selecting sentences with high scores based on the neural network's evaluation.
* **Multi-document extractive summarization**

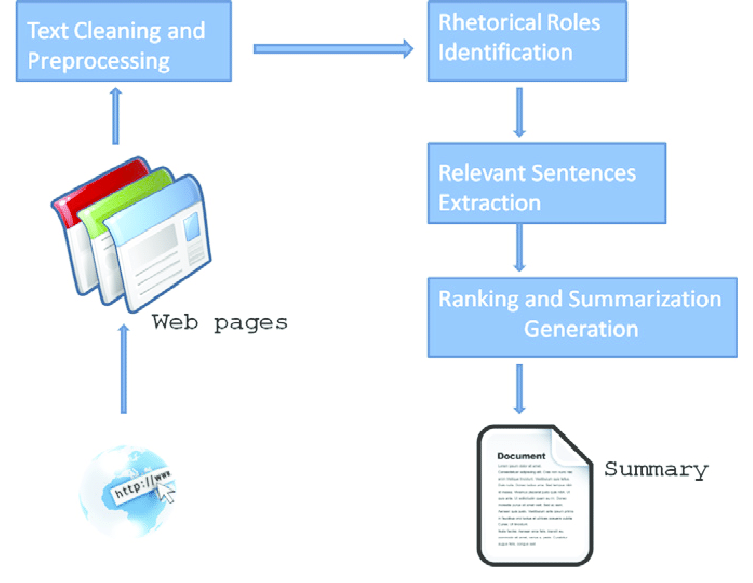
Multi-document extractive summarization is a technique used to create concise summaries by extracting important phrases or sections from multiple source documents. Unlike single-document summarization, which focuses on condensing information within a single document, multi-document summarization takes into account information from a collection of related papers. This approach aims to provide a comprehensive overview by considering multiple perspectives and sources of information. By selecting key phrases or sections from various sources, multi-document summarization helps to capture the essential information and present it in a concise and informative manner.

Fig 2.7 Multi document Summarization

**CHAPTER 3**

**PROPOSED SYSTEM**

**3.1) SYSTEM SPECIFICATIONS & REQUIREMENTS**

**3.1.1) NUMPY**

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open source project.

**3.1.2) PANDA**

Panda is a python library for data analysis. It was started by Wes McKinney in 2008.It has functions for analyzing, cleaning, exploring, and manipulating data. The name “Pandas” has a reference to both “Panel Data”, and “Python Data Analysis” .Pandas are also able to delete rows that are not relevant, or contains wrong values, like empty or NULL values.

**3.1.3) NLTK**

The Natural Language Toolkit or NLTK, is one of the premier libraries for developing Natural Language Processing (NLP) models. It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning.

**3.1.4) PYTHON**

Python is an interpreted, object-oriented & high-level programming language developed by Guido van Rossum. It was originally released in 1991.Python is a general-purpose language, meaning it can be used to create a variety of different programs and isn’t specialized for any specific problems. Python runs on an interpreter system, meaning that code can be executed as soon as it is written.

**3.1.5) VISUAL STUDIO CODE**

Visual studio code, also commonly referred to as a VS Code, is a source-code editor made by Microsoft with the Electron framework, for windows, Linux and macOS. Visual Studio Code is a free coding editor that helps us to start coding quickly. Any programming language can be coded in VS code. Visual Studio Code has support for many languages including python, java , C++, Javascript, and more.

**3.1.6) RE (REGULAR EXPRESSION)**

The Python "re" module provides regular expression support. In Python a regular expression search is written as: match = re. search(pat, str) The re.search() method takes a regular expression pattern and a string and searches for that pattern within the string. The’ re’ package is a built-in module in Python that provides functions for working with regular expressions, which are powerful tools for pattern matching and text manipulation.

**3.1.7) WARNING**

The "warnings" module provides a way to manage and control warning messages that are issued during the execution of a Python program. In NLP, the "warnings" package can be useful when working with external libraries or APIs. By using the "warnings" package, you can control how these warning messages are handled, whether to display them, ignore them, or raise an exception.

**3.1.8) WORD\_TOKENIZE**

word\_tokenize is a function in Python that splits a given sentence into words using the NLTK library. It is a common pre-processing step in NLP tasks where text data needs to be processed at the word level. One popular tool used for word tokenization in NLP is the Natural Language Toolkit (NLTK), which is a library for Python. NLTK provides a method called ’ word\_tokenize()’ that tokenizes a text string into a list of words.

**3.1.9) TEXTBLOB**

TextBlob is a popular Python library for natural language processing (NLP). It provides a simple and intuitive API for common NLP tasks such as noun phrase extraction, sentiment analysis, language translation, and more. TextBlob is built on top of NLTK (Natural Language Toolkit) and offers an easy-to-use interface for NLP operations.

**3.1.10) WORDCLOUD**

Word clouds are a visualization technique commonly used in natural language processing (NLP) to display the most frequent words or terms in a text corpus. It is great for visualizing unstructured text data and getting insights on trends and patterns.

**3.1.11) STOPWORDS**

Stop words are a set of commonly used words in any language. For example, in English, “the”, “is”, & “and”, would easily qualify as stop words. In NLP, stop words are used to eliminate unimportant words, allowing applications to focus on the important words instead.

**3.2) ALGORITHM**

1. Begin

2. Import necessary libraries:

2.1.numpy (as np): For numerical operations

2.2.pandas (as pd): For data manipulation

2.3.warnings: For handling warnings

2.4.re: For regular expression operations

2.5.nltk: Natural Language Toolkit for text processing

2.6.nltk.tokenize: For tokenization

2.7.nltk.sent\_tokenize: For sentence tokenization

2.8.textblob: For text processing and sentiment analysis

2.9.string: For string operations

2.10.nltk.corpus.stopwords: For stopwords removal

2.11.statistics.mean: For calculating the mean

2.12.heapq.nlargest: For finding the largest elements in a list

2.13.wordcloud: For creating word clouds

3. Define 'stop words' & 'punctuations' for initializing set of stopwords and

punctuations.

4. Define 'warning' module to ignore warning messages.

5. Reads three CSV files 'articles1.csv,'articles2.csv' & 'articles3.csv into three

pandas dataframes 'df\_1','df\_2' & 'df\_3'. Compares the column names of

'df\_1' & 'df\_2' using '==' operator

5.1.Repeat the same for 'df\_2' & 'df\_3'

5.2.Creates a list 'd' containing 'df\_1','df\_2' & 'df\_3'

5.3.Concatenates the DataFrames in the list 'd' using 'pd.concat(d,keys==['x','y','z'])'

5.4.Rename the column 'content' to 'article' in the concatenated dataframe

5.5.Prints the first few rows of the concatenated DataFrame using 'df.head()'.

5.6.Prints the shape of the concatenated DataFrame using 'df.shape'

5.7.Drop the 'Unnamed: 0' column from the concatenated dataframe

5.8.Print the first few rows of the modified DataFrame.

6. import 'seaborn' and 'matplotlib' libraries for data visualization.

6.1.It aims to create two count plots to visualize the distribution of publications and articles according to the year.

6.2.Set the figure size for the plots.

6.3.Set the font scale and style for seaborn.

6.4.Create the first count plot for the distribution of publications.

6.5.Rotate the x-axis labels by 45 degrees for better readability.

6.6.Set the labels and title for the first plot.

6.7.Replace a specific value in the 'year' column of the DataFrame 'df'

6.8.Create the second count plot for the distribution of articles by year

6.9.Set the labels and title for the second plot.

7. Analyse a dataframe 'df' that contains the column 'author'

7.1. calculate the frequency count of each unique value in the 'author'

column of the DataFrame 'df'.

7.2. sets the font scale, style, and grid style for the seaborn library using

'sns.set(font\_scale=1,style='whitegrid')'

7.3. selects the top 80 most frequent authors from the DataFrame using 'df\_author=df.author.value\_counts().head(80)'.

7.4. create a 'barplot' function to create a horizontal barplot. The x-axis represents the count of authors, and the y-axis represents the author names.

7.5. Add labels and titles to the x and y axis

7.6. Adjust the plot appearance using various functions

7.7. Replace the contractions

7.8. Define 'contractions\_dict', it maps certain contractions to their expanded forms.

8. import the 're' module for regular expressions.

8.1.Compile a regular expression pattern ' contractions\_re' using 're.compile'.

8.2. Define a function name called 'cleanhtml' that takes a single parameter 'raw\_HTML'

8.3. Compile another regular expression pattern 'cleanr' to match and remove HTML tags using the '<.\*?>' pattern

8.4. use 're.sub' to substitute the matches of 'cleanr' in 'raw\_ html' with an empty string, effectively removing the HTML tags.

8.5. store the result in 'cleantext'

8.6. return the cleaned text as output of the function.

9. Define a function named 'expand\_contractions' that takes two

parameters:'s' & 'contractions\_ dict'.

9.1. Define an inner function named 'replace' that takes a single parameter 'match'.

9.2. inside 'replace' return the expansion of the matched contraction 'match.group(0)' from the ' contractions\_dist' .

9.3.Define a function named 'preprocessing' that takes a single

parameter 'article'.

9.4. converted the article to lowercase

9.5. remove the HTML tags using ' cleanhtml'

9.6. use regular expression 're' for removing email addressses & URLs

9.7.Replace the non-breaking space character with a regular space character.

9.8. expand the contractions in the article using 'expand\_contractions'.

9.9. Return the preprocessed 'article' as output of the function.

10.Apply a lambda function to the 'article' series for removing possessive forms of words and reducing consecutive spaces.

10.1. Extends the preprocessing steps for the 'article' data by removing punctuation and stop words.

10.2. It applies lambda functions and regular expressions to achieve these modifications.

10.3. Store the result back in the ' article' series.

11. Analyze the frequency and score of sentences in an article using

'word\_frequency' & 'sentence\_score'.

12.Using 'summary(sentence\_score\_Owo)',it calculates the number of sentences to include in the summary by taking 25% of the total number of sentences. It selects the top-scoring sentences using the 'nlargest' function and appends them as a summary to the 'summary list'.

12.1.calculates the word frequencies using the ' word\_frequency' function.

12.2.calculates sentence scores using the 'sent\_token' function.

12.3.Generates the summarized article using the 'summary' function.

12.4.The last part of the code calls the 'article\_summarize' function with a subset of article dataframe.

12.5. Then it prints the actual length of the first article, displays the full article text, prints the length of the summarized article, displays the summarized article, and finally calls the 'word\_cloud' function to generate and display the word cloud visualization.

13.1) Import the necessary libraries:

13.1.1) 'newspaper'-library for scraping articles from websites.

13.1.2) 'reportlab'-library for generating PDF documents.

13.1.3) 'sumy'-library for text summarization.

13.2) Import specific modules from the libraries:

13. 2.1) 'article' class from 'newspaper' library to extract information from newspaper articles.

13.2.2) 'SimpleDocTemplate' , 'paragraph' and 'spacer' classes from 'reportlab.lib.styles' module for creating PDF documents.

13.2.3) 'getSampleStyleSheet' function from 'reportlab.lib.styles' module for defining document styles.

13. 2.4) 'plaintextParser', 'Tokenizer' & 'LsaSummarizer' classes from the 'sumy.parsers.plaintext' & 'sumy.nlp.tokenizers' & 'sumy.summarizers.lsa' modules respectively for text summarization.

13.3) Perform text summarization by initialising an newspaper article.

13.3.1) use a 'parse()' method to extract the article's information.

13.3.2) initialise a 'LsaSummariser' object.

13.3.3) use the 'analyze()' method of the 'LsaSummarizer' to compute the summary.

13.3.4) Get the summarized sentences using the 'get\_summary()' method.

13.4) Generate a PDF document by initialising 'SimpleDocTemplate' object with the desired output file name and page size.

13. 4.1) Create a list of paragraphs and spacers with the summarized sentences and styles.

13.4.2) Build the document using the 'build()' method of the 'SimpleDocTemplate' object.

14) End.